PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2002-082922
(43)Date of publication of application : 22.03.2002

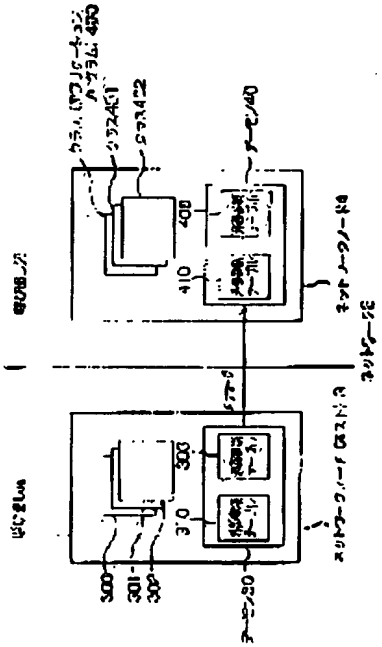
(51)Int.Cl. G06F 15/16
G06F 9/44
G06F 9/46

(21)Application number : 2001-180675
(22)Date of filing : 05.03.1996

(71)Applicant : DIGITAL VISION LABORATORIES CORP
(72)Inventor : MAEKAWA HIROTOSHI
SAITO TAKAYUKI
CHIBA TETSUHISA

(54) PARALLEL DISTRIBUTION PROCESSING METHOD

(57)Abstract:
PROBLEM TO BE SOLVED: To provide a parallel distribution processing method capable of quickly and efficiently calling a function between systems.
SOLUTION: At the time of calling the function of a class 402 of a network node 4 by a class(module) 302 of a network node 3, the network node 3 acquires the function identification code(numeric value) of the function from an external reference table 310, and outputs the function identification code with an argument as a message to the network node 4. The network node 4 acquires the execution address of the function from an internal reference table 406 by using the inputted function identification code as a key, and executes the function.



LEGAL STATUS

[Date of request for examination] 14.06.2001
[Date of sending the examiner's decision of rejection] 11.10.2005
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]
[Date of final disposal for application]
[Patent number]
[Date of registration]
[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（A）　(11)特許出願公開番号
特開2002－82922
(P2002－82922A)
(43)公開日　平成14年３月22日(2002.3.22)

(51)Int.Cl.'		識別記号	FI		テーマコード'(参考)
G06F	15/16	620	G06F 15/16	620S	5B045
	9/44	530	9/44	530M	5B098
	9/46	360	9/46	360B	

審査請求　有　請求項の数1　OL　（全 13 頁）

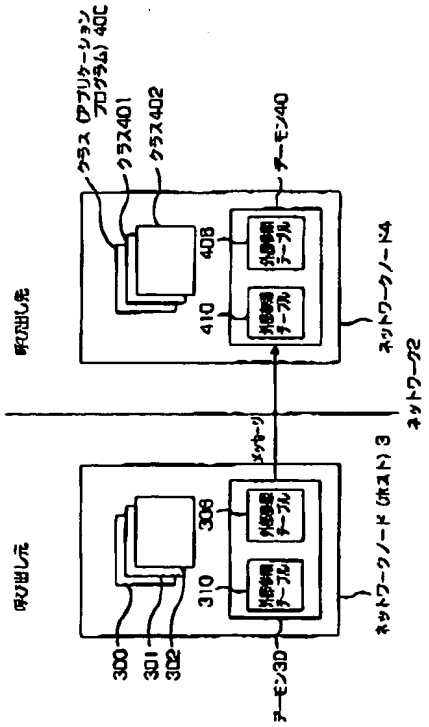
(21)出願番号　特願2001－180675(P2001－180675)
(62)分割の表示　特願平8－47833の分割
(22)出願日　平成８年３月５日(1996.3.5)

(71)出願人　396001380
株式会社ディジタル・ビジョン・ラボラトリーズ
東京都港区赤坂七丁目３番37号
(72)発明者　前川　博俊
東京都港区赤坂七丁目３番37号　株式会社ディジタル・ビジョン・ラボラトリーズ内
(72)発明者　斎藤　隆之
東京都港区赤坂七丁目３番37号　株式会社ディジタル・ビジョン・ラボラトリーズ内
(74)代理人　100094053
弁理士　佐藤　隆久

最終頁に続く

(54)【発明の名称】　並列分散処理方法

(57)【要約】
【課題】　システム相互間での機能呼び出しを高速かつ効率的に行うことができる並列分散処理方法を提供する。
【解決手段】　ネットワークノード３のクラス（モジュール）３０２において、ネットワークノード４のクラス４０２の関数（機能）の呼び出しを行うときに、ネットワークノード３は、外部参照テーブル３１０から当該関数の機能識別コード（数値）を得て、この機能識別コードが引数と共にメッセージとしてネットワークノード４に出力する。ネットワークノード４では、入力した機能識別コードをキーとして内部参照テーブル４０６から当該機能の実行アドレスを得て、当該機能を実行する。



【特許請求の範囲】

【請求項1】ネットワークを介して接続された複数のネットワークノードの各々で、機能についての一連の手続きを記述したプログラムを動作させ、一のプログラムが、他のプログラムあるいは他のプログラムに記述された手続きによって規定された機能を呼び出して実行する並列分散処理方法において、

前記複数のネットワークノード上の各々でデーモンが動作し、

前記デーモンは、

当該デーモンと同じネットワークノード上で動作する前記プログラムが規定する機能のうち他のプログラムから呼び出される機能について、当該機能の機能識別番号と当該機能の実行アドレスとの対応を示す内部参照テーブルと、

当該デーモンと同じネットワークノード上で動作する前記プログラムが呼び出しを行う他のプログラムの機能について、当該機能の機能識別子と当該機能の機能識別番号との対応を示す外部参照テーブルとを有し、

前記ネットワークノードは、前記並列分散処理の初期化時、あるいは、前記プログラムが追加された時に、前記機能の呼び出し関係に基づいて、前記内部参照テーブルおよび前記外部参照テーブルを生成あるいは更新し、

第1のネットワークノードが、前記機能の呼び出し元のプログラムに基づいて、呼び出しを行う機能の機能識別子をキーとして、当該呼び出し元のプログラムと同じネットワークノード上で動作する前記デーモンの前記外部参照テーブルから対応する機能識別番号を得て、当該機能識別番号を含むメッセージを、前記機能の呼び出し先のプログラムが動作する第2のネットワークノードに送り、

前記第2のネットワークノードが、前記呼び出し先のプログラムに基づいて、前記第1のネットワークノードから送られたメッセージに含まれる前記機能識別番号をキーとして、当該呼び出し先のプログラムと同じネットワーク上で動作する前記デーモンの前記内部参照テーブルから当該呼び出された機能の実行アドレスを得て、当該実行アドレスに基づいて、当該呼び出された機能を実行し、

前記プログラムがオブジェクト指向プログラミングのクラスである場合に、当該クラスの内部変数を共有変数とし、当該共有変数へのアクセスを、当該共有変数を内包する前記クラスへの前記機能呼び出しによって実現する並列分散処理方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、ネットワークワイドなマルチメディア並行処理を行う並列分散処理方法に関する。

【0002】

【従来の技術】近年、マルチメディア情報を利用および流通させる情報提供利用サービスアプリケーションであるＩＳＭ(Information Super-Market)の開発が活発に行われている。ＩＳＭでは、機能モジュール（クラス）間の対話性およびモジュール間構成の柔軟性を高めるために、ネットワークワイドなマルチメディア並行処理を効率的に行うアプリケーション実行環境が求められる。これに応えようとするのが、並列分散実行支援システムである。従来の並列分散実行支援システムでは、モジュール間の機能呼び出しは、文字列で示されるモジュール名と機能名とを用いて行う。ここで、機能呼び出しとは、あるモジュールで行われる処理において、他のモジュールにおいて備えてある機能を呼び出して用いることをいう。

【0003】並列分散支援システムの代表的なものに、ＣＯＲＢＡ(Common Object RequestBroker Architecture)－ＯＲＢ(Object Request Broker)がある。このＣＯＲＢＡ－ＯＲＢでは、あるモジュールが他のモジュールの機能を呼び出すにあたって、文字列で示される機能名と引数が呼び出し元モジュールから呼び出し先モジュールに転送される。呼び出し先モジュールには、文字列で示される機能名と当該機能を実行するプログラムの実行アドレスとの対応表が備えてあり、呼び出し先モジュールは、その対応表を検索することによって実行アドレスを取得し、呼び出された機能を実行する。

【0004】

【発明が解決しようとする課題】しかしながら、前述した従来のＣＯＲＢＡ－ＯＲＢでは、機能の呼び出し先において、文字列で示される機能名をキーとして、テーブルを参照して実行アドレスを特定することから、その実行部を特定するコストが大きく、機能呼び出しに伴う処理の効率が悪いという問題がある。

【0005】また、ＣＯＲＢＡ－ＯＲＢでは、モジュールの識別名の規約とその管理方法については規定されていない、すなわち、システム全体での名前空間管理は行われていない。従って、ユーザは、モジュール相互間で名前の衝突などが生じないように注意して、モジュールの識別名などを決定しなければならず、ユーザの負担が大きいという問題がある。

【0006】また、ＣＯＲＢＡ－ＯＲＢでは、機能呼び出しに伴う引数の形式（フォーマット）は、コンパイル時に別途定義しておく必要がある，このとき、定義できる引数のデータ構造は構造体および配列などの単純なものに限られ、リスト構造などの複雑なデータ構造は用いることができないという問題がある。

【0007】また、ＣＯＲＢＡ－ＯＲＢでは、呼び出し先モジュールは予め存在（登録）している必要がある。また、基本的に、呼び出し機能および引数の形式をコンパイル時に定義しておく必要があり、任意の呼び出し先のモジュールを動的にダウンロード（追加登録）して使

用しようとすると、作業が煩雑になるという問題がある。また、ＣＯＲＢＡ－ＯＲＢでは、機能呼び出しを行う際の同期方式が異なると、異なるプログラミングインターフェース（ＡＰＩ）を用いる必要があり、プラグラムの記述が複雑になるという問題がある。

【０００８】本発明は、上述した従来技術に鑑みてなされ、機能呼び出しを高速かつ効率的に行える並列分散処理方法を提供することを目的とする。また、本発明は、ユーザによるモジュールの識別名の管理負担を軽減できる並列分散処理方法を提供することを目的とする。また、本発明は、機能呼び出しにおいて、多様な形式の引数を用いることができる並列分散処理方法を提供することを目的とする。また、本発明は、煩雑な作業を経ることなく、任意の呼び出し先のモジュールを動的にダウンロード（追加登録）して使用できる並列分散処理方法を提供することを目的とする。

【０００９】

【課題を解決するための手段】上述した従来技術の問題点を解決し、上述した目的を達成するために、本発明の並列分散処理方法は、ネットワークを介して接続された複数のネットワークノードの各々で、機能についての一連の手続きを記述したプログラムを動作させ、一のプログラムが、他のプログラムあるいは他のプログラムに記述された手続きによって規定された機能を呼び出して実行する並列分散処理方法において、前記複数のネットワークノード上の各々でデーモンが動作し、前記デーモンは、当該デーモンと同じネットワークノード上で動作する前記プログラムが規定する機能のうち他のプログラムから呼び出される機能について、当該機能の機能識別番号と当該機能の実行アドレスとの対応を示す内部参照テーブルと、当該デーモンと同じネットワークノード上で動作する前記プログラムが呼び出しを行う他のプログラムの機能について、当該機能の機能識別子と当該機能の機能識別番号との対応を示す外部参照テーブルとを有し、前記ネットワークノードは、前記並列分散処理方法の初期化時、あるいは、前記プログラムが追加された時に、前記機能の呼び出し関係に基づいて、前記内部参照テーブルおよび前記外部参照テーブルを生成あるいは更新し、第１のネットワークノードが、前記機能の呼び出し元のプログラムに基づいて、呼び出しを行う機能の機能識別子をキーとして、当該呼び出し元のプログラムと同じネットワークノード上で動作する前記デーモンの前記外部参照テーブルから対応する機能識別番号を得て、当該機能識別番号を含むメッセージを、前記機能の呼び出し先のプログラムが動作する第２のネットワークノードに送り、前記第２のネットワークノードが、前記呼び出し先のプログラムに基づいて、前記第１のネットワークノードから送られたメッセージに含まれる前記機能識別番号をキーとして、当該呼び出し先のプログラムと同じネットワーク上で動作する前記デーモンの前記内部参照テーブルから当該呼び出された機能の実行アドレスを得て、当該実行アドレスに基づいて、当該呼び出された機能を実行し、前記プログラムがオブジェクト指向プログラミングのクラスである場合に、当該クラスの内部変数を共有変数とし、当該共有変数へのアクセスを、当該共有変数を内包する前記クラスへの前記機能呼び出しによって実現する。

【００１０】また、後述する実施形態には、以下の発明が開示されている。第１の発明は、ネットワークを介して接続された複数のネットワークノードの各々で、機能についての一連の手続きを記述したプログラムを動作させ、一のプログラムが、他のプログラムあるいは他のプログラムに記述された手続きによって規定された機能を呼び出して実行する並列分散処理方法において、前記複数のネットワークノード上の各々でデーモンが動作し、前記デモードは、当該デモーンと同じネットワークノード上で動作する前記プログラムが規定する機能のうち他のプログラムから呼び出される機能について、当該機能の機能識別番号と当該機能の実行アドレスとの対応を示す内部参照テーブルと、当該デモーンと同じネットワークノード上で動作する前記プログラムが呼び出しを行う他のプログラムの機能について、当該機能の機能識別子と当該機能の機能識別番号との対応を示す外部参照テーブルとを有し、前記ネットワークノードは、前記並列分散処理の初期化時、あるいは、前記プログラムが追加された時に、前記機能の呼び出し関係に基づいて、前記内部参照テーブルおよび前記外部参照テーブルを生成あるいは更新し、第１のネットワークノードが、前記機能の呼び出し元のプログラムに基づいて、呼び出しを行う機能の機能識別子をキーとして、当該呼び出し元のプログラムと同じネットワークノード上で動作する前記デーモンの前記外部参照テーブルから対応する機能識別番号を得て、当該機能識別番号を含むメッセージを、前記機能の呼び出し先のプログラムが動作する第２のネットワークノードに送り、前記第２のネットワークノードが、前記呼び出し先のプログラムに基づいて、前記第１のネットワークノードから送られたメッセージに含まれる前記機能識別番号をキーとして、当該呼び出し先のプログラムと同じネットワーク上で動作する前記デーモンの前記内部参照テーブルから当該呼び出された機能の実行アドレスを得て、当該実行アドレスに基づいて、当該呼び出された機能を実行し、前記プログラムがオブジェクト指向プログラミングのクラスである場合に、当該クラスの内部変数を共有変数とし、当該共有変数へのアクセスを、当該共有変数を内包する前記クラスへの前記機能呼び出しによって実現する並列分散処理方法。

【００１１】第２の発明は、前記機能識別子は、前記機能識別番号が格納されているアドレスを示す第１の発明の並列分散処理方法。第３の発明は、前記機能識別子は、前記呼び出し元のプログラムにおいてのみ有効な番

号を示す第１の発明に記載の並列分散処理方法。第４の発明は、前記機能識別子は、文字を含んで表現される機能名を示す第１の発明に記載の並列分散処理方法。

【００１２】第５の発明は、前記第１のネットワークノードは、前記呼び出し元のプログラムに基づいて、前記第２のネットワークノードに送る引数を、当該引数のデータ構造を論理形式表現した引数に変換し、当該変換後の引数を前記第２のネットワークノードに出力し、前記第２のネットワークノードは、前記第１のネットワークノードから入力した引数のデータ構造を前記呼び出し元における当該引数のデータ構造と同じデータ構造に変換した後に、当該引数を使用する第１の発明から第４の発明のいずれかに記載の並列分散処理方法。

【００１３】第６の発明は、前記第１のネットワークノードは、前記呼び出し元のプログラムに基づいて、呼び出し先のプログラムに前記機能識別番号を出力した後、前記第２のネットワークノードから当該機能の実行結果を入力するまで、その処理を停止する第１の発明から第５の発明のいずれかに記載の並列分散処理方法。

【００１４】第７の発明は、前記第１のネットワークノードは、前記呼び出し元のプログラムに基づいて、呼び出し先のプログラムに前記機能識別番号を出力した後、当該機能の実行結果が必要となったときに、当該機能の実行結果をまだ入力していない場合に、その処理を停止する第１の発明から第５の発明のいずれかに記載の並列分散処理方法。

【００１５】第８の発明は、前記第１のネットワークノードは、前記呼び出し元のプログラムに基づいて、前記第２のネットワークノードに前記機能識別番号を出力した後、当該機能の実行結果を入力しない第１の発明から第５の発明のいずれかに記載の並列分散処理方法。

【００１６】第９の発明は、前記第１のネットワークノードは、前記呼び出し元のプログラムに基づいて、呼び出し先のプログラムに前記機能識別番号を出力した後、所定時間経過しても当該機能の実行結果を入力しない場合に、エラー処理プログラムを起動する第１の発明から第７の発明のいずれかに記載の並列分散処理方法。

【００１７】第１０の発明は、前記第１のネットワークノードは、前記呼び出し元のプログラムに基づいて、前記メッセージに送信時刻を含め、前記第２のネットワークノードは、前記送信時刻に基づいて、呼び出された機能の実行時刻を決定する第１～第９の発明のいずれかに記載の並列分散処理方法。

【００１８】第１１の発明は、前記第１のネットワークノードは、前記呼び出し元のプログラムに基づいて、前記第２のネットワークノードに複数のメッセージを送る場合に、これらのメッセージに対応する複数の機能呼び出しを、予め決められた優先順位あるいはメッセージにおいて指定された優先順位に従って順に行う第１～１０の発明のいずれかに記載の並列分散処理方法。

【００１９】第１２の発明は、前記第１のネットワークノードおよび前記第２のネットワークノードは、　新たなプログラムが登録されたときに、前記デーモンに基づいて、そのプログラムが呼び出し先および呼び出し元となる機能について解析し、その解析結果に基づいて、前記内部参照テーブルおよび外部参照テーブルを更新する第１～１１の発明のいずれかに記載の並列分散処理方法。

【００２０】第１３の発明は、前記プログラムおよび前記機能は、それぞれオブジェクト指向プログラミングにおけるクラスおよびメソッドである第１～１２の発明のいずれかに記載の並列分散処理方法。

【００２１】第１４の発明は、前記プログラムがクラスであり、当該クラスの内部変数を共有変数として用いる場合に、前記共有変数の読み取りまたは変更を、当該クラスのメソッドで規定する第１～１３の発明のいずれかに記載の並列分散処理方法。

【００２２】第１５の発明は、ネットワークを介して接続された複数のネットワークノードを有し、前記複数のネットワークノードの各々で、機能についての一連の手続きを記述したプログラムを動作させ、一のプログラムが、他のプログラムあるいは他のプログラムに記述された手続きによって規定された機能を呼び出して実行する並列分散処理システムにおいて、前記複数のネットワークノード上の各々でデーモンが動作し、前記デモードは、当該デモーンと同じネットワークノード上で動作する前記プログラムが規定する機能のうち他のプログラムから呼び出される機能について、当該機能の機能識別番号と当該機能の実行アドレスとの対応を示す内部参照テーブルと、当該デモーンと同じネットワークノード上で動作する前記プログラムが呼び出しを行う他のプログラムの機能について、当該機能の機能識別子と当該機能の機能識別番号との対応を示す外部参照テーブルとを有し、前記ネットワークノードは、前記並列分散システムの初期化時、あるいは、前記プログラムが追加された時に、前記機能の呼び出し関係に基づいて、前記内部参照テーブルおよび前記外部参照テーブルを生成あるいは更新し、第１のネットワークノードが、前記機能の呼び出し元のプログラムに基づいて、呼び出しを行う機能の機能識別子をキーとして、当該呼び出し元のプログラムと同じネットワークノード上で動作する前記デーモンの前記外部参照テーブルから対応する機能識別番号を得て、当該機能識別番号を含むメッセージを、前記機能の呼び出し先のプログラムが動作する第２のネットワークノードに送り、前記第２のネットワークノードが、前記呼び出し先のプログラムに基づいて、前記第１のネットワークノードから送られたメッセージに含まれる前記機能識別番号をキーとして、当該呼び出し先のプログラムと同じネットワーク上で動作する前記デーモンの前記内部参照テーブルから当該呼び出された機能の実行アドレスを

得て、当該実行アドレスに基づいて、当該呼び出された機能を実行する並列分散処理システム。

【００２３】第１６の発明は、前記機能識別子は、前記機能識別番号が格納されているアドレスを示す第１５の発明に記載の並列分散処理システム。

【００２４】第１７の発明は、前記機能識別子は、前記呼び出し元のプログラムにおいてのみ有効な番号を示す第１５の発明に記載の並列分散処理システム。

【００２５】第１８の発明は、前記機能識別子は、文字を含んで表現される機能名を示す第１５の発明に記載の並列分散処理システム。第１９の発明は、前記第１のネットワークノードは、前記呼び出し元のプログラムに基づいて、前記第２のネットワークノードに送る引数を、当該引数のデータ構造を論理形式表現した引数に変換し、当該変換後の引数を前記第２のネットワークノードに出力し、前記第２のネットワークノードは、前記第１のネットワークノードから入力した引数のデータ構造を前記呼び出し元における当該引数のデータ構造と同じデータ構造に変換した後に、当該引数を使用する第１５～１８の発明のいずれかに記載の並列分散処理システム。第２０の発明は、前記第１のネットワークノードは、前記呼び出し元のプログラムに基づいて、呼び出し先のプログラムに前記機能識別番号を出力した後、前記第２のネットワークノードから当該機能の実行結果を入力するまで、その処理を停止する第１５～１９の発明のいずれかに記載の並列分散処理システム。第２１の発明は、前記第１のネットワークノードは、前記呼び出し元のプログラムに基づいて、呼び出し先のプログラムに前記機能識別番号を出力した後、当該機能の実行結果が必要となったときに、当該機能の実行結果をまだ入力していない場合に、その処理を停止する第１５～１９の発明のいずれかに記載の並列分散処理システム。第２２の発明は、前記第１のネットワークノードは、前記呼び出し元のプログラムに基づいて、前記第２のネットワークノードに前記機能識別番号を出力した後、当該機能の実行結果を入力しない第１５～１９の発明のいずれかに記載の並列分散処理システム。　第２３の発明は、前記第１のネットワークノードは、前記呼び出し元のプログラムに基づいて、呼び出し先のプログラムに前記機能識別番号を出力した後、所定時間経過しても当該機能の実行結果を入力しない場合に、エラー処理プログラムを起動する第１５～２１の発明のいずれかに記載の並列分散処理システム。第２４の発明は、前記第１のネットワークノードは、前記呼び出し元のプログラムに基づいて、前記メッセージに送信時刻を含め、前記第２のネットワークノードは、前記送信時刻に基づいて、呼び出された機能の実行時刻を決定する第１５～２３の発明のいずれかに記載の並列分散処理システム。第２５の発明は、前記第１のネットワークノードは、前記呼び出し元のプログラムに基づいて、前記第２のネットワークノードに複数のメッセージを送る場合に、これらのメッセージに対応する複数の機能呼び出しを、予め決められた優先順位あるいはメッセージにおいて指定された優先順位に従って順に行う第１５～２４の発明のいずれかに記載の並列分散処理システム。第２６の発明は、前記第１のネットワークノードおよび前記第２のネットワークノードは、　新たなプログラムが登録されたときに、前記デーモンに基づいて、そのプログラムが呼び出し先および呼び出し元となる機能について解析し、その解析結果に基づいて、前記内部参照テーブルおよび外部参照テーブルを更新する第１５～２５の発明のいずれかに記載の並列分散処理システム。第２７の発明は、前記プログラムおよび前記機能は、それぞれオブジェクト指向プログラミングにおけるクラスおよびメソッドである第１５～２６の発明のいずれかに記載の並列分散処理システム。第２８の発明は、前記プログラムがクラスであり、当該クラスの内部変数を共有変数として用いる場合に、前記共有変数の読み取りまたは変更を、当該クラスのメソッドで規定する第１５～２７の発明のいずれかに記載の並列分散処理システム。

【００２６】

【発明の実施の形態】以下、本発明の実施形態に係わる並列分散実行支援システムについて説明する。本実施形態に係わる並列分散実行支援システムは、例えば、ＩＳＭなどの実行環境を支援し、機能モジュール間の対話性およびモジュール間構成の柔軟性を高め、ネットワークワイドなマルチメディア並行処理を可能とするものである。

【００２７】図１は、本実施形態に係わる並列分散実行支援システム１の概念図である。図１に示すように、並列分散実行支援システム１は、ネットワークノード３、４がネットワーク２を介して通信を行い、一定の処理を行う。ネットワークノード３，４としては、例えば、ホストコンピュータなどが用いられる。ネットワークノード３では、デーモン３０およびクラス（モジュール）３００，３０１，３０２などのプログラムが動作する。ここで、クラスとは、具体的には、例えば、アプリケーションプログラムをいう。

【００２８】デーモン３０は、後述するように、内部参照テーブル３０６および外部参照テーブル３１０を有する。内部参照テーブル３０６は、機能識別コードとその機能が記憶されたアドレスとの対応を示すテーブルである。内部参照テーブル３０６は、ネットワークノード４からネットワークノード３に機能呼び出しを行うときに、ネットワークノード４から入力したメッセージに含まれる機能識別コードから、呼び出された機能のクラス３００～３０２におけるアドレスを得るために用いられる。

【００２９】外部参照テーブル３１０は、機能識別子としての機能識別コードが記憶されたアドレスと機能識別

コードとの対応を示すテーブルである。外部参照テーブル３１０は、ネットワークノード３からネットワークノード４に機能呼び出しを行うときに、クラス３００～３０２において呼び出されたアドレスから、その機能の機能識別コードを得るために用いられる。

【００３０】また、ネットワークノード４では、デーモン４０およびクラス（モジュール）４００，４０１，４０２などのプログラムが動作する。ネットワークノード４のクラス４００～４０２、内部参照テーブル４０６および外部参照テーブル４１０は、その意味および機能は、ネットワークノード３のクラス３００～３０２、内部参照テーブル３０６および外部参照テーブル３１０と同じである。

【００３１】図１に示す並列分散実行支援システム１では、クラスの配置、すなわち、どのクラスをどのネットワークノードあるいはプロセッサに配置したかは、原則として、システムのセットアップ時に一回のみ解析（指定）される。従って、システムの実行時において、各クラスは、他のクラスの機能を呼び出す際に呼び出し先のクラスの所在は既に知っている。尚、クラスの配置は、システムの実行時に指定されることもある。

【００３２】尚、ネットワーク２には、ネットワークノード３、４に加えて他のネットワークノードが接続されていてもよい。

【００３３】以下、オブジェクト指向の計算におけるメソッド呼び出しの処理、すなわち、ネットワークノード３でクラス３０２の処理を行っているときに、ネットワークノード４のクラス４０２に備えられた機能、具体的には関数を呼び出す場合を例示して、並列分散実行支援システム１における処理について説明する。オブジェクト指向の計算におけるメソッド呼び出しの処理では、クラス内のメソッドが例えば関数などの機能である。

【００３４】

内部参照テーブルおよび外部参照テーブルの作成

並列分散実行支援システム１では、例えば、図３に示すネットワークノード３においてクラス３０２が新たに追加（登録）され、クラス３０２が、図２に示すネットワークノード４のクラス４０２の機能を呼び出している場合には、以下に示すように、図３に示すデーモン３０の外部参照テーブル３１０および図２に示すデーモン４０の内部参照テーブル４０６を作成あるいは更新する。

【００３５】先ず、内部参照テーブルの作成について説明する。ネットワークノード３，４は、例えば、システムのセットアップ時に、それらのクラスで規定された関数について、図２，図３に示す内部参照テーブル４０６，３０６を作成する。ここで、内部参照テーブル３０６は、ローカルクラステーブル３０３、ローカルメンバ関数テーブル３０４およびローカルインスタンステーブル３０５で構成される。

【００３６】また、内部参照テーブル４０６は、ローカルクラステーブル４０３、ローカルメンバ関数テーブル４０４およびローカルインスタンステーブル４０５で構成される。ここで、ローカルメンバ関数テーブル３０４、４０４は、関数の「関数ＩＤ」と、関数「Ｆｕｎｃ」の実行アドレスである「関数番地」との対応を示している。尚、ネットワークノード３，４を含むネットワークノード相互間での関数の呼び出し関係は、システムのセットアップ時およびクラス３０２が新たに追加されたときに予め解析されており、外部参照テーブル３１０および内部参照テーブル４０６を作成する段階において、関数「Ｆｕｎｃ」およびクラス４０２にはそれぞれ「関数ＩＤ（機能識別コード）」および「クラスＩＤ」が自動的に割り当てられる。

【００３７】次に、外部参照テーブルの作成について説明する。例えば、図１に示すクラス３０２がクラス４０２に備えてある関数名「Ｆｕｎｃ」という機能を呼び出している場合に、図３に示すように、ネットワークノード３は、クラス３０２のディスパッチャ３１１から、クラス４０２および関数名「Ｆｕｎｃ」を用いて、ネットワークノード４に問い合わせ３１２を出力する。

【００３８】ネットワークノード４は、この問い合わせ３１２を受けると、内部参照テーブル４０６を参照して、クラス４０２および関数「Ｆｕｎｃ」に割り付けた「クラスＩＤ」および「関数ＩＤ」を図３に示す回答メッセージ３１５に含めて、図１に示すネットワーク２を介してネットワークノード３に出力する。回答メッセージ３１５は、ネットワークノード３において、メッセージキュー（待ち行列）３１２，３１３を介してディスパッチャ３１１に出力される。

【００３９】次に、ディスパッチャ３１１は、登録処理を行い、回答メッセージ３１４に含まれる「クラスＩＤ」および「関数ＩＤ」を用いて、リモートクラステーブル３０７およびリモートメンバ関数テーブル３０８を作成する。ここで、リモートクラステーブル３０７、リモートメンバ関数テーブル３０８およびリモートインスタンステーブル３０９によって、外部参照テーブル３１０が構成される。

【００４０】内部参照テーブルおよび外部参照テーブルの作成は、ネットワークノード３とネットワークノード４とがネットワーク２を介して接続されたときに初期化過程として行われたり、処理の実行時に、ネットワークノード３，４に新たなクラス（モジュール）が動的にダウンロード（追加登録）されたときに、一度のみ自動的に行われる。

【００４１】機能呼び出し処理

以下、前述したように作成された外部参照テーブル３１０および内部参照テーブル４０６を用いて、ネットワークノード３からネットワークノード４に対して機能呼び出しを行う場合について図２～６を参照しながら説明する。ここでは、呼び出し元のクラス（オブジェクト）

が、機能呼び出しを行った後に、呼び出し先のクラスから当該機能の実行結果を入力するまで、その処理を停止する、いわゆる完全同期メッセージ送信の場合について説明する。

【００４２】例えば、図４に示すネットワークノード３のクラス３０２において処理が行われているときに、ネットワークノード４に備えてあるクラス４０２の関数「Ｆｕｎｃ」という機能を呼び出すと（図６（Ａ）に示す「Ａ１」）、クラス３０２のディスパッチャ３１１において、メッセージに含まれる引数のデータ構造が論理形式のデータ表現に変換される（図６（Λ）に示す「Ｂ１」）。当該変換は、変換プログラムを用いて、クラス３０２での内部表現データのデータ構造を解析し、論理形式のデータ表現に変換して行われる。このとき、論理形式のデータ表現は、単純なデータを組み合わせることで、リスト構造などの複雑なデータ構造を表現できる記述構文を採用している。そのため、引数として、多様なデータ構造を用いることができる。そのため、機能呼び出しの対象とできる関数の種類の範囲を拡大できる。

【００４３】次に、ディスパッチャ３１１において、関数「Ｆｕｎｃ」およびクラス４０２のアドレスが、それぞれリモートメンバ関数テーブル３０８およびリモートインスタンステーブル３０９に出力される。これによって、機能識別コードの参照が行われ、これらのアドレスに対応した「関数ＩＤ」および「クラスＩＤ」が検索される（図６（Ａ）に示す「Ｂ２」）。この検索された「関数ＩＤ」および「クラスＩＤ」は、ディスパッチャ３１１において、送信キュー３１３に登録される（図６（Ａ）に示す「Ｂ３」）。

【００４４】次に、送信キュー３１３に登録された「関数ＩＤ」および「クラスＩＤ」が、デーモン３０の送信キュー３１４に登録される（図６（Ａ）に示す「Ｃ１」）。デーモン３０において、メッセージ３２０のメッセージヘッダが「関数ＩＤ」および「クラスＩＤ」を用いて作成され、このメッセージヘッダを含むメッセージが、図５に示すネットワークノード４のデーモン４０の受信キュー４１４に、ネットワークを介して送信される（図６（Ａ）に示す「Ｃ２」）。ネットワークノード３は、ネットワークノード４から関数「Ｆｕｎｃ」の処理結果を入力するまで、実行中のプログラムの処理を停止する。このように、メッセージキュー３１４，４１４を用いることで、メッセージは、キューに蓄えられた後に、該当関数（機能）によって処理される。そのため、これらのキューによって、ある特定のクラスに対しての機能呼び出し要求が競合しても、呼び出し要求が失われることなく、順番にまたは優先順位に従って処理される。

【００４５】ネットワークノード４のデーモン４０は、ネットワークノード３からメッセージを受信すると（図６（Ｂ）に示す「Ｄ１」）、そのメッセージを図５に示す受信キュー４１４に登録する（図６（Ｂ）に示す「Ｄ２」）。次に、デーモン４０からディスパッチャ４１１にシグナルが出力される（図６（Ｂ）に示す「Ｄ３」，「Ｅ１」）。

【００４６】次に、ディスパッチャ４１１は、「クラスＩＤ」をキーとして、図５に示すローカルインスタンステーブル４０５を参照し、所定のインスタンスを呼び出す（図６（Ｂ）に示す「Ｅ２」）。ディスパッチャ４１１は、所定のインスタンスが呼び出し可能な状態になるまで、次の処理を実行する（図６（Ｂ）に示す「Ｅ３」）。このとき、メッセージキュー４１４に設けられた通信マネージャは、メッセージ３２０に含まれる「クラスＩＤ」をキーとして、ローカルインスタンステーブル４０５が参照され、クラス４０２のインスタンスの番地を得る。これによって、クラス４０２のインスタンスが決定される。ここで、インスタンスとは、クラスの処理を実際に行うときの状態を示す実体をいう。

【００４７】尚、クラス４０２におけるインスタンスの生成は、図５に示すように、ローカルクラステーブル４０３においてクラス４０２と対応して記憶された「インスタンス生成関数の番地」を参照し、この「インスタンス生成関数の番地」にあるインスタンス生成関数４２１を実行し、これによって生成したインスタンスをローカルインスタンステーブル４０５に登録して行われる。

【００４８】次に、ディスパッチャ４１１は、メッセージのデータ構造を元のクラス３０２で用いられているデータ構造と同じものに変換する（図６（Ｂ）に示す「Ｅ４」）。

【００４９】次に、受信したメッセージ３２０に含まれる「関数ＩＤ」および「クラスＩＤ」に基づいて、ローカルメンバ関数テーブル４０４を用いて、呼び出された機能に関する関数「Ｆｕｎｃ」の「関数番地」を得る。このとき、「関数ＩＤ」および「クラスＩＤ」は数値で示されているため、その照合は高速に行われる。

【００５０】ネットワークノード４では、ディスパッチャ４１１において得られた関数「Ｆｕｎｃ」の「関数番地」、通信マネージャによって決定されたインスタンスおよびディスパッチャ４１１において変換されて得られた内部表現データ４２０を用いて、クラス４０２における処理が行われる（図６（Ｂ）に示す「Ｆ」）。この処理結果は、返却値として、図６（Ｂ）に示す「Ｅ５」、「Ｄ４」、「Ｄ５」および図６（Λ）に示す「Ｃ３」、「Ｃ４」、「Ｃ５」、「Ｂ４」、「Ｂ５」および「Ａ２」の処理を行うことで、図３に示す呼び出し元のネットワークノード３のクラス３０２の処理に送られる。これによって、機能呼び出し処理が完了する。

【００５１】ここで、「Ｅ５」は返却値のデータ表現の変換、「Ｄ４」は送信キューへの返却値の登録、「Ｄ５」はメッセージの送信、「Ｃ３」はメッセージの受信、「Ｃ４」は受信キューへの登録、「Ｃ５」はクラス

３０２へのシグナル送信、「Ｂ４」はシグナル受信、「Ｂ５」は返却値のデータ表現の変換および「Ａ２」は次の処理の継続実行を意味する。ネットワークノード３は、ネットワークノード４から関数「Ｆｕｎｃ」の処理結果を入力すると、停止していたプログラムの処理を再開する。

【００５２】以上説明したように、並列分散実行支援システム１によれば、関数などの機能の呼び出しは、クラス相互間での機能呼び出し関係および呼び出される機能についての数値で示される機能識別コードなどの関係を示した内部参照テーブルおよび外部参照テーブルを用いて行われるため、呼び出された機能の実行部のアドレスを高速に特定することができる。また、並列分散実行支援システム１によれば、機能呼び出しが行われるクラスおよび関数のアドレスについて、システムが自動的にユニークな機能識別コードを付与する。そのため、ユーザ（プログラマ）はクラスおよび関数のアドレスが競合することを回避するために多大な労力を費やす必要はなく、ユーザの負担が軽減される。

【００５３】また、並列分散実行支援システム１によれば、クラスの追加に応じて内部参照テーブルおよび外部参照テーブルを動的に作成することから、必要なクラス（モジュール）を動的に指定したネットワークノードにダウンロードすることが可能となる。また、並列分散実行支援システム１によれば、機能呼び出しの引数と処理結果（返り値）の形式（フォーマット）をコンパイル時に定義しておく必要がないため、機能呼び出しの柔軟性を高めることができる。

【００５４】ところで、プロセッサなどのシステムの資源を効率よく使用するために、ユーザ（プログラマ）が意識しないレベルでプログラムを細かい並行分散実行単位に自動的に分割するプログラミングシステムは存在する。これに対して、並列分散実行支援システム１は、ユーザがアプリケーションプログラムを大きな機能単位で並行分散モジュールに分割し、その並行分散モジュールの配置を明示的に指定できるものであり、大きなアプリケーションシステムの構築を可能とするものである。

【００５５】次に、ネットワークノード３からネットワークノード４に対して機能呼び出しを行う場合についての他の例を説明する。ここでは、呼び出し元のクラス（オブジェクト）が、機能呼び出しを行った後に、呼び出し先のクラスから当該機能の実行結果を入力しない、いわゆる非同期メッセージ送信の場合について説明する。

【００５６】図７は、非同期メッセージ送信の場合のタイミングチャートである。すなわち、図４に示すネットワークノード３のクラス３０２において処理が行われているときに、ネットワークノード４に備えてあるクラス４０２の関数「Ｆｕｎｃ」を非同期のメッセージ送信で呼び出すと（図７（Ａ）に示す「Ｃ１」）、ネットワークノード３のクラス３０２、ディスパッチャ３１１およびデーモン３０において、図７（Ａ）に示す「Ｈ１」、「Ｈ２」、「Ｈ３」、「Ｉ１」、「Ｉ２」および「Ｇ２」の処理が実行される。

【００５７】図７（Ａ）に示す「Ｈ１」、「Ｈ２」、「Ｈ３」、「Ｉ１」および「Ｉ２」の処理は、前述した図６（Ａ）に示す「Ｂ１」、「Ｂ２」、「Ｂ３」、「Ｃ１」および「Ｃ２」の処理に対応する。ただし、図７に示す例では、クラス３０２は、呼び出しを行ったネットワークノード４のクラス４０２の機能の処理結果を待つことはせず、処理「Ｉ２」の後に、次の処理「Ｇ２」を実行する。

【００５８】一方、ネットワークノード４では、デーモン４０、ディスパッチャ４１１およびクラス４０２において、図７（Ｂ）に示す「Ｊ１」、「Ｊ２」、「Ｊ３」、「Ｋ１」、「Ｋ２」、「Ｋ３」、「Ｋ４」および「Ｌ」の処理を実行する。ここで、図７（Ｂ）に示す「Ｊ１」、「Ｊ２」、「Ｊ３」、「Ｋ１」、「Ｋ２」、「Ｋ３」、「Ｋ４」および「Ｌ」の処理は、図６（Ｂ）に示す「Ｄ１」、「Ｄ２」、「Ｄ３」、「Ｅ１」、「Ｅ２」、「Ｅ３」、「Ｅ４」および「Ｆ」の処理と、基本的には同じであるが、「Ｆ」の処理結果は、ネットワークノード３には送信されない。

【００５９】次に、上述した本実施形態に係わる並列分散実行支援システムの概念についてまとめる。図８に示すように、ネットワークノード９８は、オブジェクト１０１ａにおけるメソッド１０１ｂにおいて、ネットワークノード９９のメソッド１０５ａのメソッド呼び出しを行っている。このメソッド呼び出しの処理において、メソッド本体１０６のＩＤを示すメソッド１０１ｂのメソッドＩＤがメッセージに含められる。また、引数１０１ｃは、そのデータ構造が論理形式に変換された引数１０１ｃ１となった後にメッセージに含められる。また、オブジェクト１０１ａからの呼び出しキーに基づいて、外部参照表１０２が参照され、オブジェクトＩＤがメッセージに含められる。メッセージは、ネットワークノード９８からネットワークノード９９に送信される。

【００６０】ネットワークノード９９では、ネットワークノード９８から送信されたオブジェクトＩＤをキーとして、内部参照表１０３を用いて、呼び出し対象となっているメソッド１０５ａのメソッド本体１０６が実行される。このとき、論理変換された引数１０１ｃ１が、オブジェクト１０１ａにおける引数のデータ構造と同じデータ構造に変換され、この変換された引数１０５ｂを用いて、メソッド本体１０６の処理が実行される。

【００６１】本発明は、上述した実施形態には限定されない。例えば、上述した実施形態では、本発明をオブジェクト指向プログラミングにおけるメソッドの呼び出しに適用した場合について例示したが、本発明は、例えば、リモート関数呼び出しにも適用できる。リモート関

数呼び出しは、リモートのアプリケーションなどのモジュールに存在する関数（機能）を呼び出すものである。

【0062】また、本発明は、あるクラスの内部変数を分散モジュール（クラス）群間の共有変数（大域変数）として使用する場合にも適用できる。この場合には、共有変数となる変数を内包するクラスの機能として、共有変数の値の読み取り、変更などの機能をプログラムする。このとき、共有変数へのアクセスは、全て、その変数を内包するクラスへの機能呼び出しによって行われるので、自然に、その共有変数へのアクセスの排他制御が実現される。すなわち、共有変数は、インスタンスの中に抱え込まれており、当該共有変数を読み出しおよび書換えて操作するには、当該インスタンスの機能を呼び出す必要がある。これによって、インスタンスが共有変数そのものとして振る舞う。このとき、共有変数をオブジェクト（クラス）とし、その共有変数についての処理をメソッド（関数）とする。

【0063】また、上述した実施形態では、呼び出し元のモジュールとしてのネットワークノード3において、機能識別子として機能識別コードが記憶されたアドレスを例示したが、機能識別子としては、文字などで示される機能名あるいは呼び出し元モジュールにおいてのみ有効な番号を用いれば、さらに高速な機能呼び出し処理を行うことができる。

【0064】また、ネットワークノード3，4相互間で入出力されるメッセージを蓄えるキューは、入力順に出力する他、所定の優先順位に基づいて、メッセージの出力順序を決定するようにしてもよい。すなわち、機能呼び出しの優先順位を付けるようにしてもよい。この優先順位は、予めシステムとして定められていてもよいし、メッセージに属性情報として含められていてもよい。

【0065】また、メッセージ送出側において、送出時刻をメッセージに含め、受信側において、その時刻に基づいて、そのメッセージで特定される機能の実行時刻を決定するようにしてもよい。また、メッセージを受信するキューは、各インスタンス（クラス）毎に存在してもよいし、複数のインスタンス（クラス）で共有するようにしてもよい。

【0066】また、本発明は、ネットワークノード3は、メッセージをネットワークノード4に出力した後に、ネットワークノード4からの処理結果を必要になるまで処理を続行し、必要となった時点で処理結果をまだ受け取っていない場合にのみ、実行中のプログラムの処理を停止するようにしてもよい。このとき、ネットワークノード3は、ネットワークノード4から、既に送ったメッセージについての処理結果を、設定時間を経過しても得られない場合に、自動的にエラー処理プログラムを起動するようにしてもよい。また、本発明は、ネットワークノード3は、メッセージをネットワークノード4に出力した後に、ネットワークノード4からの処理結果を要求しないようにしてもよい。

【0067】

【発明の効果】以上説明したように、本発明の並列分散処理方法によれば、機能呼び出しを高速かつ効率的に行うことができる。また、本発明の並列分散処理方法によれば、ユーザによるモジュールの識別名の管理負担を軽減できる。また、本発明の並列分散処理方法によれば、機能呼び出しにおいて、多様な形式（データ構造）の引数を用いることができる。また、本発明の並列分散処理方法によれば、煩雑な作業を経ることなく、任意の呼び出し先のモジュールを動的にダウンロード（追加登録）して使用できる。

【図面の簡単な説明】

【図1】図1は、本発明の実施形態に係わる並列分散実行支援システムの概念図である。

【図2】図2は、図1に示す並列分散実行支援システムにおいて、内部参照テーブルを作成する手法について説明するための図である。

【図3】図3は、図1に示す並列分散実行支援システムにおいて、外部参照テーブルを作成する手法について説明するための図である。

【図4】図4は、図1に示す並列分散実行支援システムにおいて、機能呼び出しを行う際の、呼び出し元のネットワークノードにおける処理を説明するための図である。

【図5】図5は、図1に示す並列分散実行支援システムにおいて、機能呼び出しを行う際の、呼び出し先のネットワークノードにおける処理を説明するための図である。

【図6】図6は、完全同期メッセージ送信を行う場合の図1に示す並列分散実行支援システムのタイミングチャートである。

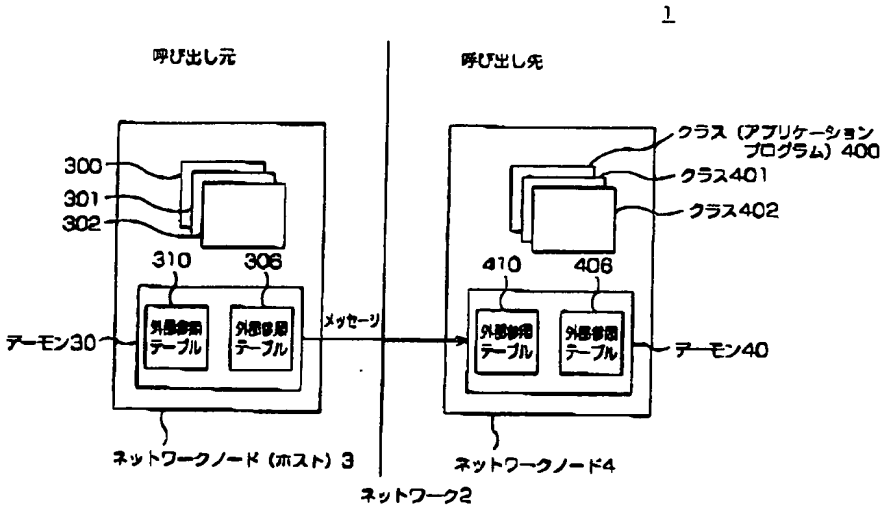
【図7】図7は、非同期メッセージ送信における場合の図1に示す並列分散実行支援システムのタイミングチャートである。

【図8】図8は、本発明の並列分散実行支援システムの概念図である。

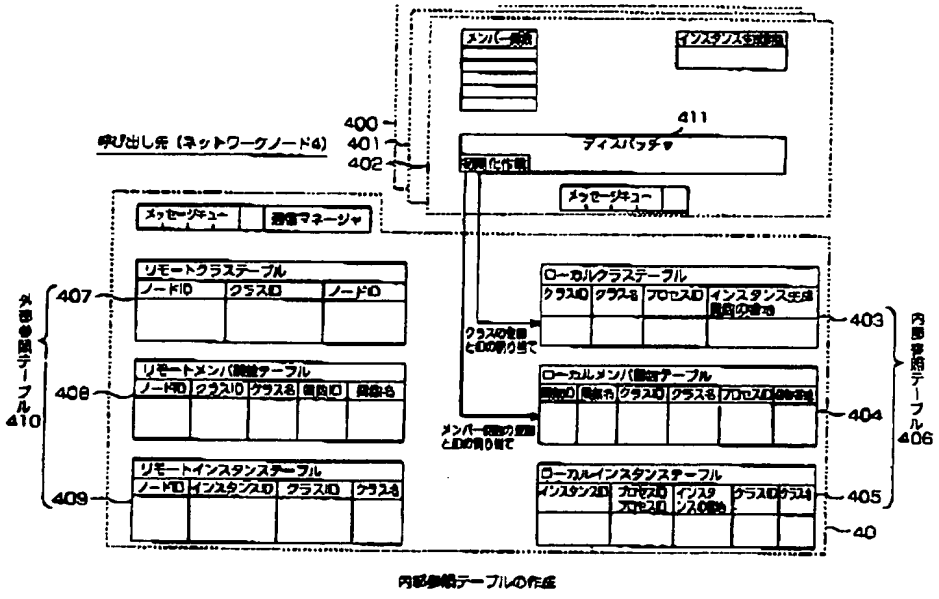
【符号の説明】

1…並列分散実行支援システム
2…ネットワーク
3，4…ネットワークノード
30，40…デーモン
300，301，302，400，401，402…クラス
306，406…内部参照テーブル
310，410…外部参照テーブル

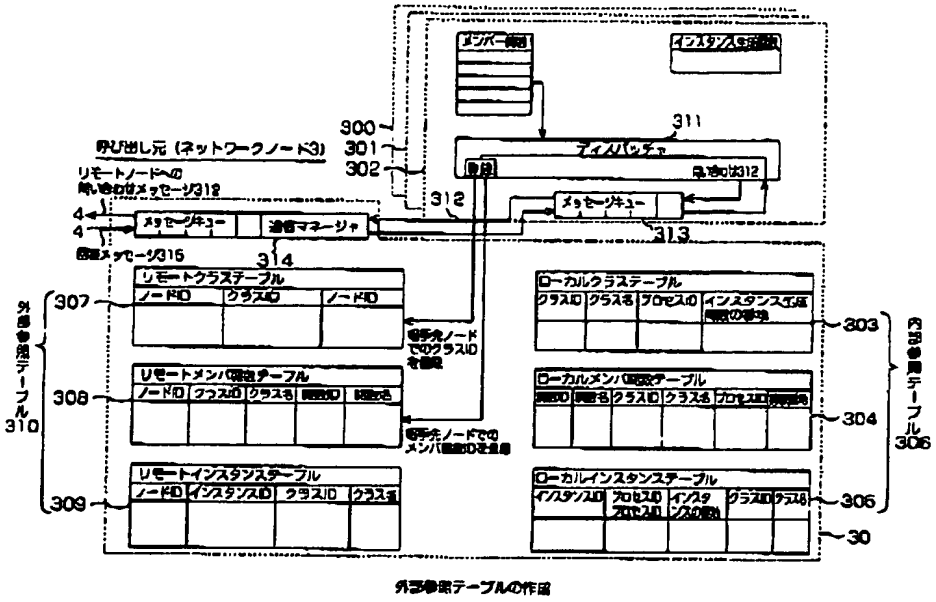
【図１】


1
呼び出し元
呼び出し先
300
301
302
310
306
外部参照テーブル
外部参照テーブル
デーモン30
メッセージ
ネットワークノード（ホスト）3
クラス（アプリケーションプログラム）400
クラス401
クラス402
410
406
外部参照テーブル
外部参照テーブル
デーモン40
ネットワークノード4
ネットワーク2

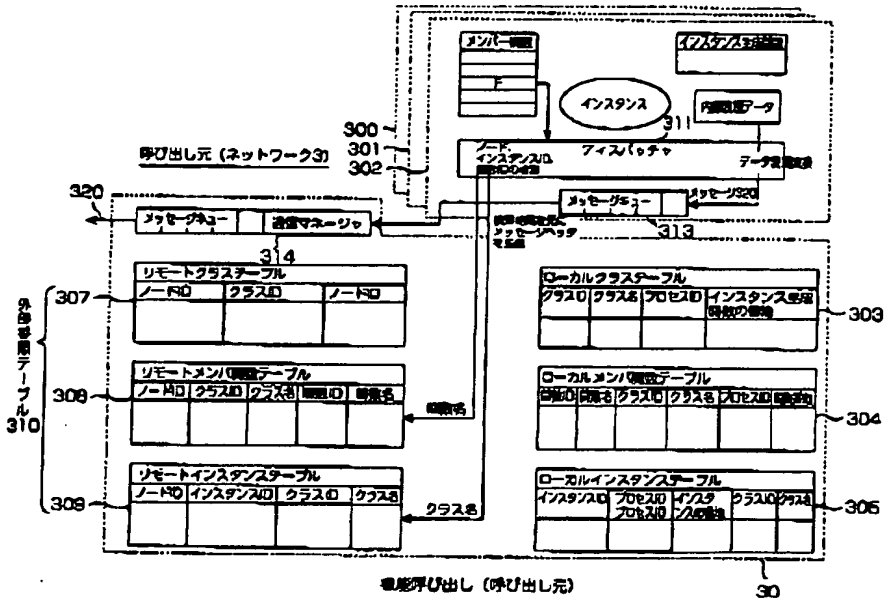
【図２】


呼び出し先（ネットワークノード4）
400
401
402
411
メッセージキュー
通信マネージャ
リモートクラステーブル
ノードID
クラスID
ノードID
407
リモートメンバ関数テーブル
408
リモートインスタンステーブル
409
外部参照テーブル
410
ローカルクラステーブル
クラスID
クラス名
プロセスID
403
ローカルメンバ関数テーブル
404
ローカルインスタンステーブル
405
内部参照テーブル
406
40
内部参照テーブルの作成

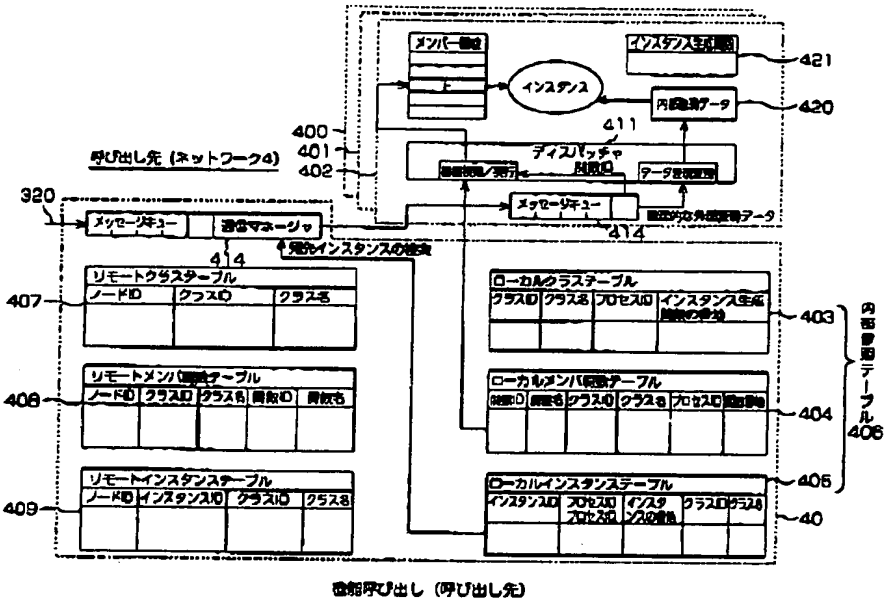
【図3】


300
301
302
311
312
313
314
307
308
309
303
304
306
30
310
外部参照テーブル
内部参照テーブル
呼び出し元 (ネットワークノード3)
リモートクラステーブル
ローカルクラステーブル
リモートメンバ関数テーブル
ローカルメンバ関数テーブル
リモートインスタンステーブル
ローカルインスタンステーブル
メッセージキュー
通信マネージャ
ディスパッチャ
メンバー関数
インスタンス生成関数
外部参照テーブルの作成

【図4】


300
301
302
311
313
314
307
308
309
303
304
306
30
310
320
呼び出し元 (ネットワーク3)
リモートクラステーブル
ローカルクラステーブル
リモートメンバ関数テーブル
ローカルメンバ関数テーブル
リモートインスタンステーブル
ローカルインスタンステーブル
メッセージキュー
通信マネージャ
ディスパッチャ
インスタンス
内部参照データ
メッセージID
メッセージヘッダ
関数名
クラスID
メンバー関数
インスタンス生成関数
外部参照テーブル
内部参照テーブル
機能呼び出し (呼び出し元)

【図５】



【図６】

(A)

時間 ⟶ 呼び出し元のモジュール

クラス	A1																			A2
ディスパッチャ		B1	B2	B3	スリープ												スリープ	B4	B5	
デーモン				C1	C2											C3	C4	C5		

(B)

時間 ⟶ 呼び出し先のモジュール

デーモン						D1	D2	D3					D4	D5						
ディスパッチャ								E1	E2	E3										
											E4		E5	スリープ						
クラス												F								

【図７】

(A)

時間→　呼び出し元のモジュール

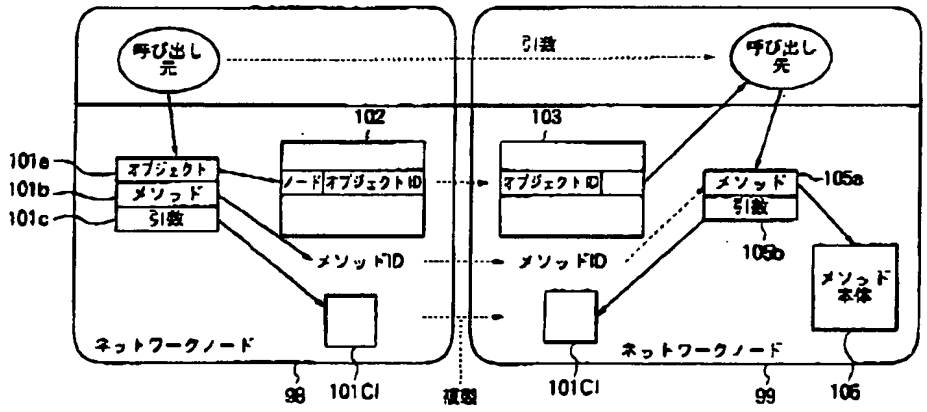
クラス	G1				G2							
ディスパッチャ		H1	H2	H3								
デーモン				I1	I2							

(B)

時間→　呼び出し先のモジュール

デーモン						J1	J2	J3				
ディスパッチャ								K1	K2	K3	K4	
クラス												L

【図８】



フロントページの続き

(72)発明者　千葉　哲央
東京都港区赤坂七丁目３番37号　株式会社
ディジタル・ビジョン・ラボラトリーズ内

Ｆターム(参考)　5B045 GG01 GG11
5B098 GA07 GC00